Obsługa kart pamięci i generowanie dźwięku przy pomocy modułu ARD WAV Shield

Wstęp

Dzisiejszy „wykład" postanowiłem podzielić na 2 bloki, bo choć cały czas będziemy korzystać z modułu Wav Shield, to zrealizujemy z jego wykorzystaniem aż 2 różne zagadnienia a mianowicie:

1. Obsługa kart pamięci jako nośników danych – czyli zapis i odczyt danych, np. w celu zapisywania wartości pomiarów itp.
2. Współpraca kart pamięci i przetwornika cyfrowo-analogowego w celu odtwarzania plików muzycznych z karty.

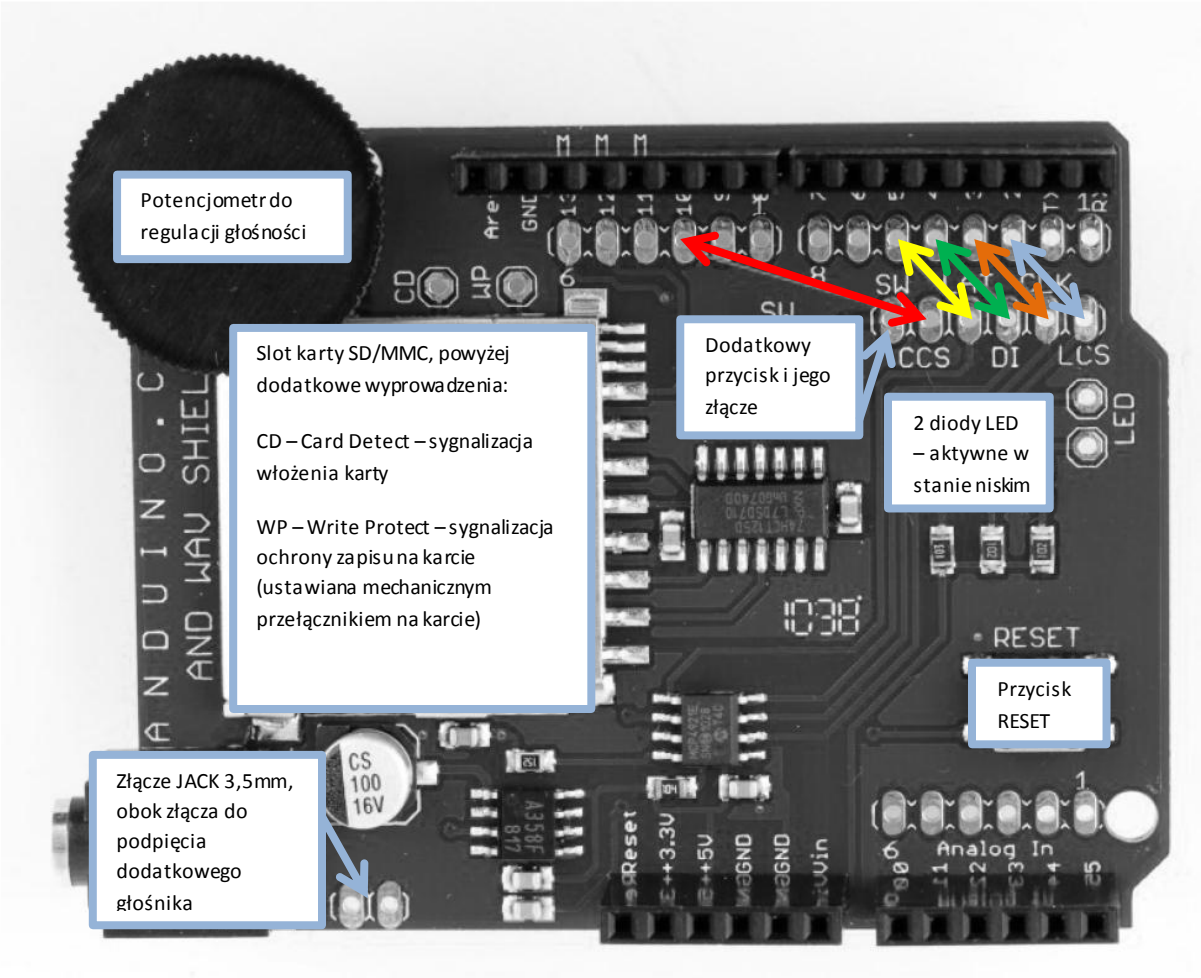
Podczas omawiania tych zagadnień będziemy korzystać z 2 różnych bibliotek, gdyż jedna z nich jest specjalnie dedykowana i optymalizowana do odtwarzania dźwięku, druga zaś posiada funkcje przystosowane do odczytu i zapisu dowolnych danych do plików. To tyle słowem wstępu, a teraz czas na rozpoczęcie pracy.

Przygotowanie modułu do pierwszego uruchomienia

Moduł Wav Shield wymaga przed pierwszym użyciem montażu dostarczonych elementów przewlekanych, co nie powinno sprawiać większych trudności nawet początkującym korzystającym z prostych lutownic transformatorowych. Kolejną czynnością, którą należy wykonać, jest podłączenie odpowiednich wyprowadzeń układów i karty SD do wyprowadzeń naszej płytki. Mamy tutaj pełną dowolność w kwestii podłączenia pinu CS[1] karty SD oraz wszystkich używanych do komunikacji wyprowadzeń przetwornika cyfrowo-analogowego, jednak z racji tego, iż będziemy korzystać z gotowych bibliotek musimy podłączyć te wyprowadzenia tak, jak zostały wykorzystane w bibliotekach. Poprawne połączenie[2] pokazuje rysunek na następnej stronie. Opisałem na nim także niektóre elementy, a bardziej szczegółowych informacji możemy uzyskać z instrukcji obsługi oraz schematu elektronicznego tego modułu.

[1] CS (z ang. Chip Select) – pin wyboru i aktywacji układu, zwykle utrzymywanie tego pinu w stanie niskim powoduje, ze układ jest aktywny i może się komunikować, a podanie na ten pin stanu wysokiego „dezaktywuje" układ i pozwala na komunikację z innymi urządzeniami na tych samych liniach

[2] Połączenia najlepiej wykonać za pomocą izolowanych i przyciętych na odpowiednią długość kawałków drutu miedzianego.



Obsługa karty SD, jako nośnika danych

Zgodnie z obietnicą na początek zajmiemy się wykorzystaniem karty SD jako nośnika danych dowolnego przeznaczenia. Właściwie do tej części kursu wystarczy tylko podłączenie pinu CCS do pinu 10, gdyż pozostałe linie sygnałowe karty są już na stałe podłączone do procesora (do pinów 5-3), jednak jeśli już będziemy takowe połączenia wykonywać połączenie wszystkich wyprowadzeń nie będzie sprawiało żadnego problemu[3]. Oczywiście nie musze już chyba nadmieniać, że pinów tych raczej nie powinniśmy wykorzystywać do innych celów, chyba że dokładnie wiemy co robimy[4]. Nadmienię jeszcze iż urządzenie na pewno współpracuje z kartami SD i MicroSD[5] o pojemnościach do 2GB włącznie. Większych kart nie testowałem, dlatego nie gwarantuję ich działania, tak samo jak kart SD HC. Skoro więc posiadamy już odpowiednio przygotowany sprzęt możemy napisać pierwszy program testujący działanie karty. Może on wyglądać np. tak:

[3] Choć nie będziemy mogli dowolnie sterować także na razie nie używanym pinem 3 płytki, który powinien być pozostawiony jako wejście bez podciągania lub lepiej jako wyjście ustawione w stan wysoki – to spowoduje iż przetwornik nie będzie zakłócał w żaden sposób transmisji między procesorem a kartą.
[4] Na razie, na naszym etapie nauki uznajemy te piny jako wykorzystane i nie podłączamy do nich żadnych innych urządzeń.
[5] w odpowiednim adapterze

```
#include <SD.h> //dołączamy bibliotekę do obsługi kart SD
File plik; //tworzymy obiekt, który będzie przechowywać informacje konieczne do prowadzenia operacji na plikach

char znak;

void setup(){
  Serial.begin(9600);
  Serial.print("Inicjalizacja karty SD: ");

  pinMode(10,OUTPUT);//ustawiamy pin 10, podłączony do pinu CS karty jako wyjście (aby program mógł nim poprawnie sterować)

  if(SD.begin(10)){//do funkcji begin przekazujemy także numer tego pinu
  //UWAGA! Używając kiedyś innego pinu pamiętajcie o tym, aby pin 10 ustawić mimo wszystko jako wyjście i pozostawić takowy niepodłączony, gdyż jest on i tak "zajmowany" przez kontroler SPI
  //SPI - interfejs szeregowy wykorzystywany m. in. przez karty SD i inne układy
    Serial.println("OK");
    //jeśli funkcja zwróci prawdę (logiczną 1) to oznacza to poprawną inicjalizację karty
  }else{
    Serial.println("BLAD - sprawdz karte");
    while(1);
    //w przeciwnym razie wiemy, iż nie wszystko przebiegało poprawnie - wyświetlamy stosowny komunikat i zatrzymujemy działanie programu poprzez pętlę nieskończoną.
  }

  Serial.print("Plik testowy.txt ");
  if(SD.exists("testowy.txt")){//tak sprawdzamy czy dany plik istnieje
    Serial.println("istnieje");

    ///plik = SD.open("testowy.txt", FILE_READ);//otwieramy plik do odczytu
    plik = SD.open("testowy.txt");//przy otwieraniu pliku do odczytu nie musimy tego wyraźnie zaznaczać - plik zostanie domyślnie otwarty do odczytu
    if(plik){//powinniśmy sprawdzić czy plik został poprawnie otwarty
      Serial.println("Plik otwarty do odczytu");
    }else{
      Serial.println("Plik nie został otwarty! - sprobuj ponownie");
      while(1);
      //wystąpiły jakieś błędy - zatrzymujemy program
    }

    while(plik.available()){//metoda available działa dokładnie tak samo jak w przypadku komunikacji RS232 - jeśli pozostały jakieś znaki do odczytania w pliku to funkcja zwraca wartość TRUE
      Serial.print(plik.read(), BYTE);//funkcja read
    }

    plik.close();//zamykamy plik

  }else{
    Serial.println("nie istnieje");
  }


  plik = SD.open("testowy.txt", FILE_WRITE);//teraz otwieramy plik testowy.txt w trybie zapisu.
```

```
  //jeśli plik wcześniej istniał zostanie otwarty w trybie dodawania
danych, w przeciwnym razie zostanie on utworzony na karcie SD
  if(plik){//powinniśmy sprawdzić czy plik został poprawnie otwarty
    Serial.println("Plik otwarty do zapisu, wpisz linijke tekstu: ");
  }else{
    Serial.println("Plik nie został otwarty! - sprobuj ponownie");
    while(1);
    //wystąpiły błędy - zatrzymujemy program
  }

  while(znak!=10){
   if(Serial.available()){
    znak=Serial.read();
    //plik.write(znak);//tak zapisujemy pojedyncze znaki
    plik.print(znak, BYTE);//równie dobrze możemy zastosować znane nam
polecenia print i println - ich działanie jest dokładnie takie samo jak w
przypadku komunikacji RS232
    Serial.print(znak, BYTE);//dodatkowo wyświetlamy znaki w terminalu
   }
  }
  plik.print("Program pracowal ");
  plik.print(millis()/1000.0, 2);
  plik.println(" sekund.");
  //dospisujemy jeszcze trochę tekstu

  Serial.println("Tekst zostal zapisany. Zresetuj urzadzenie aby wykonac
operacje ponownie");

  plik.close();//i pamiętamy koniecznie o zamknięciu pliku

}

void loop(){

}
```

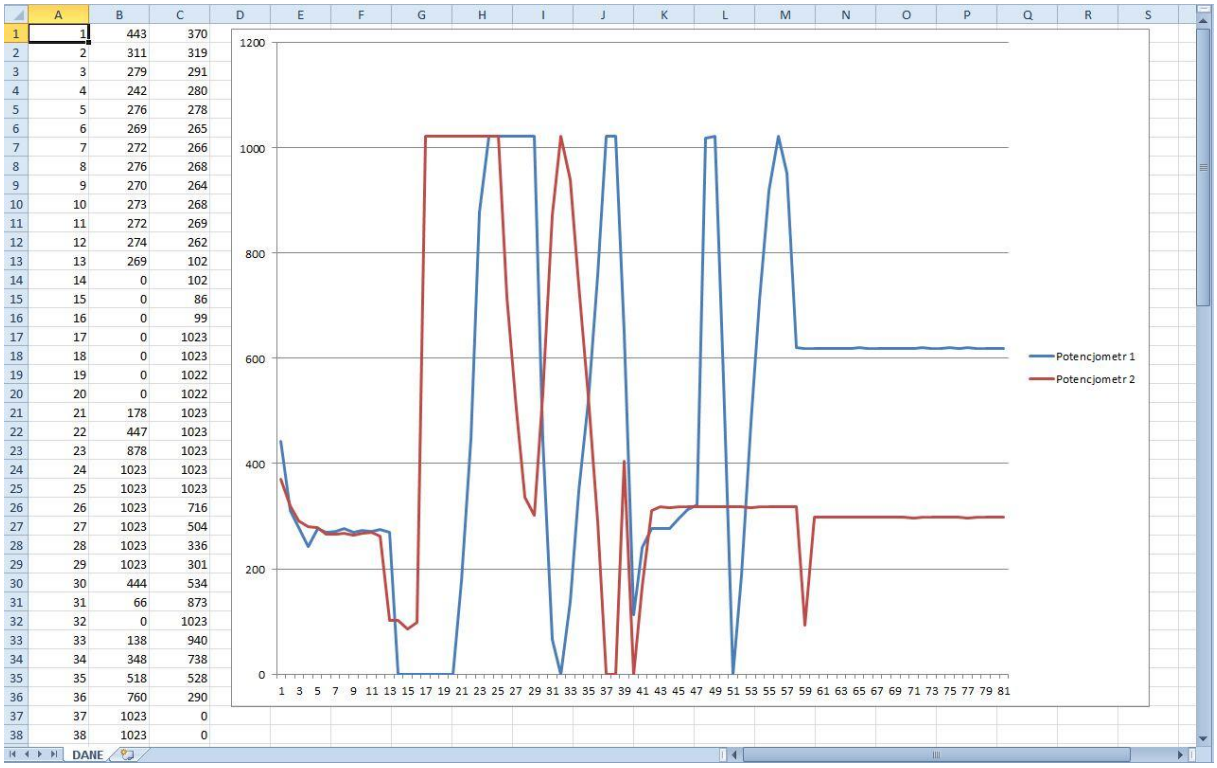
Mam nadzieję, że nie muszę już chyba wyjaśniać szerzej kwestii zapisu danych na kartę SD. No może oprócz jednego – mianowicie nazwa pliku nie może być dłuższa niż 8 znaków, a rozszerzenie nie może być dłuższe niż 3 znaki, nie wspominając już o wykorzystaniu tylko i wyłącznie liter z alfabetu łacińskiego. Jednak chciałbym jeszcze wspomnieć o paru kwestiach, a mianowicie sposobie zapisu i odczytu liczb z plików na karcie. Moim zdaniem najwygodniejszym sposobem takiego zapisu są pliki CSV (z ang. Comma Separated Values – wartości oddzielone przecinkami). Wielką zaletą takich plików jest możliwość ich szybkiego eksportu do wielu arkuszy kalkulacyjnych takich jak np. Excel®. W plikach csv każda kolejna wartość oddzielona średnikiem[6] zostanie zapisana do kolejnej komórki w danym wierszu, zaś przejście w pliku do nowej linii spowoduje przejście do następnego wiersza komórek w arkuszu. Już zapewne widzicie jak przydatne mogą okazać się te pliki w przypadku chęci narysowania np. wykresu, czy porównania różnych danych. Pierwszą kwestię zapisu danych do pliku CSV przedstawię na przykładzie odczytu danych z wejść analogowych. Osobiście podłączyłem do tych wyprowadzeń potencjometry, ale nic nie stoi na przeszkodzie, aby podpiąć także dowolny układ pomiarowy[7]. Stworzymy wtedy bardzo prosty, ale skuteczny rejestrator.

[6] liczby zmiennoprzecinkowe są zapisywane z przecinkami (przynajmniej w polskiej wersji programu Microsoft® Excel®)
[7] np. układy, które konstruowaliśmy przy okazji omawiania zagadnienia pomiaru wartości analogowych

```
#include <SD.h> //dołączamy bibliotekę do obsługi kart SD
File plik; //tworzymy obiekt, który będzie przechowywać informacje
konieczne do prowadzenia operacji na plikach
word analog;
unsigned long last_time;

void setup(){
  Serial.begin(9600);
  Serial.print("Inicjalizacja karty SD: ");

  pinMode(10,OUTPUT);

  if(SD.begin(10)){
    Serial.println("OK");
  }else{
    Serial.println("BLAD - sprawdz karte");
    while(1);
  }
    Serial.println("Rozpoczynam rejestracje, wpisz i wyslij \"a\" aby
przerwac i bezpiecznie wyjac karte");

  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
}

void loop(){

  if(millis()%1000==0){
    //dane zapisujemy co 1 sekundę - gdy reszta z dzielenia milisekund
przez 1000 wynosi 0
    //dzięki takiemu podejściu do zagadnienia... nie musimy korzystać z
instrukcji delay(); dzięki czemu możemy w czasie oczekiwania wykonywać inne
operacje
    plik=SD.open("dane.csv",FILE_WRITE);
    //otwieramy plik do zapisu
    if(plik){
      plik.print(millis()/1000,DEC);
      plik.print(";");
      Serial.print(millis()/1000,DEC);
      Serial.print(";");
      analog=analogRead(0);
      plik.print(analog,DEC);
      plik.print(";");
      Serial.print(analog,DEC);
      Serial.print(";");
      analog=analogRead(1);
      plik.print(analog,DEC);
      plik.println("");
      Serial.print(analog,DEC);
      Serial.println("");
      //zapsiujemy do pliku wartości oddzielone średnikami, te same dane
wysyłamy przez port RS232 do komputera
      plik.close();
    }else{
     Serial.println("Blad odczytu pliku - zrestartuj urzadzenie");
     while(1);
    }
  }
  if(Serial.available()){
   if(Serial.read()=='a'){
```

```
    Serial.println("Urzadzenie zatrzymane - mozesz wyjac bezpiecznie karte.
W celu ponownego uruchomienia zresetuj urzadzenie");
    while(1);
    //jeśli odebraliśmy znak a to zatrzymujemy program w celu usunięcia
karty SD z urządzenia
   }
  }


}
```

Przykładowe dane i wykres wygenerowany na ich podstawie mogą wyglądać np. tak:



	A	B	C
1	1	443	370
2	2	311	319
3	3	279	291
4	4	242	280
5	5	276	278
6	6	269	265
7	7	272	266
8	8	276	268
9	9	270	264
10	10	273	268
11	11	272	269
12	12	274	262
13	13	269	102
14	14	0	102
15	15	0	86
16	16	0	99
17	17	0	1023
18	18	0	1023
19	19	0	1022
20	20	0	1022
21	21	178	1023
22	22	447	1023
23	23	878	1023
24	24	1023	1023
25	25	1023	1023
26	26	1023	716
27	27	1023	504
28	28	1023	336
29	29	1023	301
30	30	444	534
31	31	66	873
32	32	0	1023
33	33	138	940
34	34	348	738
35	35	518	528
36	36	760	290
37	37	1023	0
38	38	1023	0

Zanim przejdziemy do odczytu danych z pliku CSV, chciałbym jeszcze zwrócić uwagę na kwestę zapisu do plików CSV danych zmiennoprzecinkowych. Wszystko wydaje się być w tej kwestii bezproblemowe, jednak Arduino zapisuje liczby z częścią dziesiętną odd zieloną przecinkiem, zaś Excel®[8] wymaga zapisu z przecinkiem. Napiszmy zatem bardzo prostą funkcję konwertującą liczbę na jej reprezentację z przecinkiem. Większość programu pozostaje bez zmian, zatem umieszczę tylko kluczowe fragmenty kodu. Pełny kod znajdziecie w plikach dołączonych do tej części kursu.

```
#include <SD.h> //dołączamy bibliotekę do obsługi kart SD
File plik; //tworzymy obiekt, który będzie przechowywać informacje
konieczne do prowadzenia operacji na plikach
word analog;
float napiecie;
//String konwersja;
```

[8] przynajmniej jego polskojęzyczna wersja

```
String konwertuj(float liczba){
  liczba*=100.0;
  //chcemy uzyskać dokładność 2 miejsc po przecinku, zatem "przesuwamy" go
o 2 miejsca
  int liczba2=liczba;
  //pozostałe miejsca dziesiętne nas nie interesują zatem konwertujemy
liczbę na całkowitą
  String konwersja;
  konwersja=String(liczba2/100);
  //na początku zapisujemy części całkowite
  konwersja+=",";//dodajemy przecinek
  konwersja+=String(liczba2%100);
  //i wypisujemy 2 miejsca dziesiętne
  return konwersja;
}

void setup(){
  //konfiguracja
 }

void loop(){
      //…sprawdzanie czasu, …
      plik.print(millis()/1000,DEC);
      plik.print(";");
      Serial.print(millis()/1000,DEC);
      Serial.print(";");
      napiecie=analogRead(0);
      napiecie*=5.0;
      napiecie/=1023;
      plik.print(konwertuj(napiecie));
      plik.print(";");
      Serial.print(konwertuj(napiecie));
      Serial.print(";");
      napiecie=analogRead(0);
      napiecie*=5.0;
      napiecie/=1023;
      plik.print(konwertuj(napiecie));
      plik.println("");
      Serial.print(konwertuj(napiecie));
      Serial.println("");
      plik.close();
      //…pozostała część obsługi rejestratora…

}
```

Teraz przyszła już kolej na odwrotną operację, czyli odczyt danych z pliku CSV. Operacja taka może przydać się nam choćby do wczytania wcześniej zapisanej konfiguracji, czy też ustawień urządzenia. Jak jednak zabierzemy się do odczytu takich danych liczbowych? Dokładnie tak samo, jak w przypadku pobierania ich poprzez terminal RS232 od użytkownika.

```
#include <SD.h> //dołączamy bibliotekę do obsługi kart SD
File plik; //tworzymy obiekt, który będzie przechowywać informacje
konieczne do prowadzenia operacji na plikach
int numery[3];
int i;
int odczyt;

void setup(){
  Serial.begin(9600);
```

```
  Serial.print("Inicjalizacja karty SD: ");
  pinMode(10,OUTPUT);
  if(SD.begin(10)){
    Serial.println("OK");
  }else{
    Serial.println("BLAD - sprawdz karte");
    while(1);
  }

  plik=SD.open("licz.csv",FILE_READ);
  i=0;
  while(plik.available()){//powtarzamy dopóki w pliku są jakieś
nieodczytane dane
    odczyt=plik.read();
    if(odczyt!=';'&&odczyt!=13&&odczyt!=10){//jeśli odczytywany znak to nie
jeden ze znaków zakończenia linii lub średnik
      numery[i]*=10;//to aktualną wartość liczby przesuwamy o jedno miejsce
      numery[i]+=odczyt-'0';//i dodajemy odczytaną cyfrę (metoda jak w
lekcji o RS232)
    }else if(odczyt==';'){//jeśli natrafiliśmy na średnik
      i++;//to przechodzimy do następnej kolumny (następnej komórki w
tabeli)
    }else if(odczyt==10){//jeśli to już koniec linii to wyświetlamy liczby
z niej odczytane
      Serial.print(numery[0], DEC);
      Serial.print("\t");//znak tabulacji spowoduje, że liczby będą ładnie
wyrównane
      Serial.print(numery[1], DEC);
      Serial.print("\t");
      Serial.println(numery[2], DEC);
      numery[0]=0;
      numery[1]=0;
      numery[2]=0;
      i=0;
      //następnie resetujemy zmienne przechowujące liczby
    }
  }

}

void loop(){

}
```

W tej chwili wykorzystanie tych wielkich możliwości zależy już wyłącznie od waszej inwencji twórczej. Na „zadanie domowe" polecam Wam tak zmodyfikować nasz kod, aby odczytywał także liczby zmiennoprzecinkowe. Mam nadzieję, że po chwili zastanowienia i przeanalizowaniu obecnego kodu nie sprawi to nikomu większych trudności.

Odtwarzanie dźwięku z karty SD

Teraz przyszła już kolej na chyba najbardziej oczekiwaną część naszego kursu – odtwarzanie dźwięków z wykorzystaniem naszej płytki i modułu. Pierwszą czynnością jaką musimy wykonać jest oczywiście przygotowanie karty SD. Na początek nadmienię tylko, że w przypadku odtwarzania plików dźwiękowych bardzo ważna jest odpowiednia szybkość ich odczytu z karty SD. Jeśli

zagłębilibyśmy się w szczegóły standardu FAT16/32 (których używamy) zobaczylibyśmy, że po usunięciu pliku dane nie są „przesuwane" na początek karty, ale zostaje po nim puste miejsce. Później, jeśli wgrywamy na kartę nowy, większy plik, zostanie on podzielony (fachowo: pofragmentowany) i wpasowany w pozostałe puste miejsca. Podczas odczytu zaś wystąpi konieczność „skakania" po tych miejscach, w których został on zapisany. Z tego powodu autorzy biblioteki dźwiękowej zalecają wgrywanie plików dźwiękowych bez ich usuwania. Jeśli zaś wcześniej już wielokrotnie usuwaliśmy i zapisywaliśmy pliki najlepiej ponownie sformatować kartę. Oczywiście nie jest to konieczne, ale w przeciwnym wypadku mogą wystąpić nieoczekiwanie zakłócenia (np. w formie trzasków).

Drugą kwestią, którą muszę poruszyć jest format samego pliku z dźwiękiem. Plik powinien być zapisany jako format *.WAV, czyli pozbawiony jakiejkolwiek kompresji. Dodatkowo plik powinien posiadać tylko jeden kanał (nagranie mono, gdyż nasz odtwarzacz jest wstanie odtwarzać tylko jeden kanał), rozdzielczość powinna wynosić 16-bit, a częstotliwość próbkowania 44100 Hz. W przeciwnym wypadku podczas próby odtwarzania dźwięku wystąpi błąd. Wszystkich tych zmian możemy dokonać np. za pomocom programu *Advanced WMA Workshop*. W Internecie możemy znaleźć bez trudu jego wersję Shareware, która przez określony czas będzie w zupełności spełniać nasze oczekiwania. Aby dokonać konwersji przechodzimy w w.w. programie do katalogu, w którym znajduje się plik z dźwiękiem, który chcemy skonwertować. Następnie klikamy na pliku PPM[9] i wybieramy *Convert to WAV*. Później przechodzimy do zakładki *WAV Settings* i wybieramy *Samplerate: 44100 Hz* oraz *Channels mode: Mono*. Następnie zatwierdzamy działanie i po chwili w naszym katalogu mamy już gotowy plik z muzyką do skopiowania na kartę. Nazewnictwo plików muzycznych rządzi się takimi samymi prawami jak zwykłych, czyli maksymalnie 8 znaków nazwy i 3 znaki rozszerzenia oraz kropka.

Zanim przedstawię Wam przykładowy program chciałbym nadmienić, iż oryginalna biblioteka jest nieco skomplikowana w obsłudze, dlatego stworzyłam specjalny zestaw funkcji ułatwiających pracę. W przyszłości wystarczy, że skopiujecie w odpowiednie miejsca deklaracje i definicje tych funkcji, aby tworzyć własne programy z wykorzystaniem dźwięku. A teraz kolej na program realizujący prosty odtwarzacz muzyki sterowany z komputera poprzez terminal RS232.

```
/*
Dołączenie bibliotek
*/
#include <WaveHC.h>
#include <WaveUtil.h>
/*
Zmienne i obiekty systemu odtwarzania plików - nie modyfikować
*/
SdReader card;
FatVolume vol;
FatReader root;
FatReader file;
WaveHC wave;
dir_t dirBuf;
uint8_t part;
/*
przykładowe zmienne wykorzystywane w naszym programie
UWAGA! tablica przechowująca nazwę pliku musi mieć dokładnie 13 elementów.
```

[9] Prawym Przyciskiem Myszy

```
W przeciwnym wypadku może dojść do nieprawidłowego funkcjonowania programu
lub nawet skasowania plików z karty SD
*/
uint8_t CARD_init_state;
uint8_t WAV_play_state;
char name[13];
byte opcja;

uint8_t dirNameToCharArray(uint8_t *byte_table, char *char_table);
/*
funkcja konwertująca tablicę bajtów na tablicę znaków (tekst), używana tylk
o i wyłącznie wewnetrznie
*/

#define CARD_INIT_OK  0
#define CARD_INIT_ERR 1
#define PARTITION_ERR 2
#define ROOT_OPEN_ERR 3

uint8_t init_SD(uint8_t slow_spi=0, uint8_t opt_read=1);
/*
inicjalizacja karty SD
slow_spi=0 -> używanie szybkiego SPI 8MHz
slow_spi=1 -> używanie wolnego SPI 4 MHz
opt_read=1 -> optymalizacja odczytu włączona
opt_read=0 -> optymalizacja odczytu wyłączona

funkcja zwraca:
0 -> inicjalizacja prawidłowa
1 -> błąd inicjalizacji karty
2 -> błąd otwierania partycji
3 -> błąd otwierania systemu plików
*/

#define dir_read_init(); root.rewind();
#define dir_read_init() root.rewind()
/*
przyjazna nazwa dla inicjalizacji odczytu katalogu
*/

uint8_t find_wav(char *char_table);
/*
wyszukiwanie plików WAV na karcie SD
funckja do tablicy podanej jako parametr zapisuje nazwę pliku, zwraca FALSE
, jeżeli na karcie nie ma już więcej plików WAV
należy wtedy użyć instrukcji dir_read_init();, aby rozpocząć wyszukiwanie o
d początku
*/

#define is_playing(); wave.isplaying;
#define is_playing() wave.isplaying
/*
przyjanza nazwa dla funkcji sprawdzającej, czy aktualnie jest odtwarzany ja
kiś plik dźwiękowy
*/

#define stop_wav(); wave.stop();
#define stop_wav() wave.stop()
/*
przyjazna nazwa funkcji zatrzymującej odtwarzanie
```

```
*/

#define pause_wav(); wave.pause();
#define pause_wav() wave.pause()
/*
przyjazna nazwa funkcji pauzującej odtwarzanie dźwięku
*/

#define resume_wav(); wave.resume();
#define resume_wav() wave.resume()
/*
przyjazna nazwa funkcji wznawiającej odtwzarzanie zapauzowanego dźwięku
*/

#define is_paused(); wave.isPaused();
#define is_paused() wave.isPaused()
/*
przyjazna nazwa funkcji sprawdzającej, czy dźwięk jest zapauzowany
*/

#define PLAY_OK       0
#define OPENING_ERROR 1
#define INVALID_WAV   2

uint8_t play_wav(char *char_table);
/*
funkcja ładująca dźwięk i rozpoczynająca jego odtwarzanie
tablica podana jako parametr powinna zawierać nazwę pliku do odtwarzania
funkcja zwraca:
0 -> odtwarzanie rozpoczęte poprawnie
1 -> błąd otwierania pliku (plik uszkodzony lub nie istnieje)
2 -> nieprawidłowy lub nieobsługiwany format pliku WAV
należy pamiętać, że pliki dźwiękowe muszą być przygotowane zgodnie z poniżs
zymi parametrami:
rozdzielczość: 16-bit
częstotliwosc próbkowania: 44,100 kHz
kanały: mono
*/


void setup()
{
  Serial.begin(9600);
  Serial.println("");
  Serial.println("Test odtwarzacza wav.");


  if(CARD_init_state=init_SD()){//inicjalizujemy kartę - jeden znak
równości to nie błąd:
  /*
  taki warunek zadziała następująco:
  1. zostanie wykonana funkcja init_SD()
  2. wartość , którą ona zwróci zostanie zapisana w ziemmen CARD_init_state
  3. do funkcji if zostanie przekazana wartość tej zmiennej (dla jasności -
 każda wartość różna od 0 jest traktowana jako prawda)
  Czyli całość mógłbym zapisać tak:
  CARD_init_state=init_SD();
  if(CARD_init_state){....
  */
    Serial.print("Blad inicjalizacji karty, kod: ");
```

```
    Serial.println(CARD_init_state, DEC);
    while(1);
  }

}

void loop()
{


  dir_read_init();//przewijamy katalog do początku (pliki wyszukiwane w
kolejności dodania)
  while(find_wav(name)){//powtarzamy dopóki będziemy znajdować nowe pliki,
których nazwy trafiają do tablicy name
    if(WAV_play_state=play_wav(name)){//podobnie jak przy inicjalizacji
karty, teraz sprawdzamy czy da się rozpocząć odtwarzanie pliku.
    /*
    Równie dobrze możemy wpisać stałą nazwę pliku, np:
    if(WAV_play_state=play_wav("VIVALDI.WAV")){....
    */
     Serial.print("!! Blad odtwarzania ");
     Serial.print(name);
     Serial.print(": Kod bledu: ");
     Serial.println(WAV_play_state);
    }else{
     Serial.print("-> Odtwarzam ");
     Serial.println(name);
    }
   while(is_playing()){//powtarzamy dopóki plik jest odtwarzany
    if(Serial.available()){
      opcja=Serial.read();//odczytujemy dane z RS232 i dokonujemy
odpowiedniej operacji
      switch(opcja){
         case 'q'://stop - następny utwór
           stop_wav();
           Serial.println("[] Stop");
         break;
         case 'w'://pauza/wzownienie odtwarzania
           if(is_paused()){//sprawdzamy czy odtwarzanie jest zapauzowane
czy nie
             resume_wav();
             Serial.println("-> Odtwarzanie");
           }else{
             pause_wav();
             Serial.println("|| Pauza");
           }
         break;
      }
    }
   }
  Serial.println("");
  }

}

/*
definicje funkcji wykorzystywanych przez odtwarzacz, nie zmieniaj nic poniż
ej
*/
uint8_t dirNameToCharArray(uint8_t *byte_table, char *char_table){
```

```
  uint8_t is_wav=true;
  for(int i=0;i<=11;i++){
    byte znak=*(byte_table);
    if(znak!=32)*(char_table++)=(char)znak;
    if(i==7)*(char_table++)='.';
    byte_table++;
    if(i==7&&*(byte_table)!='W')is_wav=false;
    if(i==8&&*(byte_table)!='A')is_wav=false;
    if(i==9&&*(byte_table)!='V')is_wav=false;
  }
  *(char_table)=0;
  return is_wav;
}

uint8_t init_SD(uint8_t slow_spi, uint8_t opt_read){
if (!card.init(slow_spi)) return CARD_INIT_ERR;
card.partialBlockRead(opt_read);
for (part = 0; part < 5; part++) if (vol.init(card, part)) break;
if (part == 5) return PARTITION_ERR;
if (!root.openRoot(vol)) return ROOT_OPEN_ERR;
return CARD_INIT_OK;
}

uint8_t find_wav(char *char_table){
  while(root.readDir(dirBuf)){
    if(dirNameToCharArray(dirBuf.name,char_table))return true;
  }
  return false;
}

uint8_t play_wav(char *char_table){
  if(!file.open(root, char_table)) return OPENING_ERROR;
  if(!wave.create(file)) return INVALID_WAV;
  wave.play();
  return PLAY_OK;
}
```

Podsumowanie

Oczywiście zachęcam do stworzenia własnego odtwarzacza, w którym sterowanie będzie realizowane poprzez przyciski, a dane o utworze będą wyświetlane na wyświetlaczu, tak aby urządzenie pracowało niezależnie od komputera. Możecie także wykorzystać ten sprzęt do odtwarzania dźwięków, czy ostrzeżeń, a nawet skonstruować mówiący zegarek z termometrem – wszystko zależy od Waszej inwencji twórczej!